Assignment 1

CS-GY 6033 INET Fall 2024

Due date: 11:55pm on Sept 27th on **Gradescope**.

Instructions:
Below you will find the questions which make up your homework. They are to be written out (or typed!) and handed in online via Gradescope before the deadline.

Question 1: Asymptotic Notation

.

(a) 8 points Rank the following functions in order (non-decreasing) of their asymptotic growth. Next to each function, write its big-Theta value, (ie. write the correct $\Theta(g(n))$ next to each function but you are not required to *prove* the big-Theta value).

$$\log(n^3+2n),\ \sqrt{n^3(\log n)+n},\ (2^{10}+6)(2^n+3^n),\ \frac{n^2\log n+n}{n+\log n},\ 2^{3n+1},\ \frac{\sqrt{n\log n+1}}{n^2+1},\ 8^{n/3+1},$$

$$\log_3 2^n,\ 2^{\log_3 n},\ 2^{\log_2 n+1},\ (\log(n/2))^2$$

(b) 6 points Determine if each of the following statements are true or false. If the statement is false, provide a counter example. If the statement is true, justify the statement using the formal definitions from class.

1. If $f(n)$ is $O(n)$, does this imply that $f(n)$ is also $\Theta(n)$?
2. If $f(n)$ is $O(n^3)$, does this imply that $f(n)$ is also $O(3^n)$?
3. If $f(n)$ is $O(n^2)$, does this imply that $f(n)$ is also $O(n)$?

(c) 16 points For each of the following $f(n)$, show that $f(n)$ is $\Theta(g(n))$ for the correct function $g(n)$. Prove your result using the definitions from class, justifying your statement is true for all $n \geq k$. (provide the value of k).

- $f(n) = n^2 - \frac{\sqrt{n}}{\log n} + n(\log n)^2$
- $f(n) = \frac{3n^3-n}{2n+\log n} + 2^{10}$
- $f(n) = 2^n \cdot n + n^5 \log n - 1.5^n$
- $f(n) = \sqrt{n^3+1} + n^2\sqrt{n+1}$

Question 2: Operations and Runtime

(a) 12 points Below is the link to the description of a comparison-based sorting algorithm. Watch the video and make sure you understand how the algorithm works.

`https://stream.nyu.edu/media/simplesort/1_lra8y1yh`

Your job:

- Write the pseudo-code (non-recursive) for this algorithm. You **must** use the skeleton below, called SimpleSort(A, s, f) which sorts array A between indices s and f. The final sorted array is in array $L[1, \dots, n]$

```
SimpleSort(A,s,f)
    $n =$ ________        Set this to the number of elements in $A$
    Initialize array $L[1, \dots, n]$        Array to store the two subarrays to be merged.
    last = ________        The value of the last element in current sorted list of L[]
    lastindex = ________        The index last element in the current sorted list of L[]
    end = 0
    while lastindex $\neq$ ________
        for $i = s$ to $f$
            if $A[i] >$ last        The element at A[i] should be added to L[]
                lastindex ++
                last = ________
                ____________
        Merge($L$, 1, ________, ________)
        end= ________
        ____________
```

- What is the worst-case number of comparisons on array A of length n? You must describe the input that causes this worst-case scenario, and justify the worst-case number of comparisons.
- What is the best-case number of comparisons made on array A of length n? You must describe the input that causes this best-case scenario, and justify the best-case number of comparisons.
- Do some research to find the name of this algorithm, and find its average runtime.

(b) 12 points Below is a non-recursive variation of MergeSort, which is intended to sort array A between indices s and f. Note that it makes reference to the Merge procedure from week 2. You should remind yourself of the specific parameters of the Merge algorithm.

```
MergeRuns(A,s,f)
    $e = s$
    while $e < f$
        if $A[e] < A[e+1]$
            $e++$
        else break
    while $e < f$
        $e2 = e + 1$
        while $e2 < f$
            if $A[e2] < A[e2+1]$
                $e2++$
            else break
        Merge($A, s, e, e2$)
        $e = e2$
```

- Execute the procedure on $[3, 4, 5, 1, 6, 2]$. Ensure you show the state of the array during the execution of the procedure.
- Is this a correct sorting algorithm?

- Let $T(n)$ be the best-case runtime of this procedure. Express $T(n)$ is a function and show that it is $O(n)$.
- Suppose the input array A has n elements which are sorted in decreasing order. In this case, show that the number of comparisons is $O(n^2)$.

Question 3: Recurrences and recursive sorting

(a) 3 points Recall that the pseudo-code from class for *MergeSort* required external space as part of the merge step. These types of algorithms are referred to as sorting algorithm that do *not* run in-place. Do some research to determine if there is a known version of MergeSort that runs in-place, and if so, what its best-known runtime is.

(b) 8 points Suppose we want to re-write the MergeSort algorithm so that it splits the array into three subarrays each of size approximately $n/3$. Your job is to write the pseudo-code for this new version of MergeSort. Ensure that you have a valid base case! (you can check this by testing it on input of size n = 1,n = 2. You will have to write a new version of the Merge procedure, which merges three sorted arrays. You can call this procedure MergeThree$(A, s, q, q2, f)$, which merges sorted subarrays $A[s \dots q], A[q+1 \dots q2], A[q2+1 \dots f]$.

(c) 4 points Write and justify the runtime recurrence for the above procedure, and use Master Method to show that the runtime is $\Theta(n \log n)$

(d) 8 points Consider the the pseudo-code below for the recursive algorithm MyPrint(A, s, f), which takes as input an array A, indexed between s and f.

```
MyPrint(A,s,f)
    if $2 < f - s$
        q1 = $\lfloor (2s + f)/3 \rfloor$
        q2 = $\lfloor (s + 2f)/3 \rfloor$
        MergeSort$(A, q1, q2)$
        MyPrint$(A, q1, f)$
        MyPrint$(A, s, q2)$
    else
        MergeSort$(A, s, f)$
```

Your job:

- Trace the recursive algorithm on the input $A = [5, 4, 3, 2, 1]$ indexed from $s = 1$ to $f = 5$. Ensure that you show the state of the array during the recursive calls, and detail the parameters passed at each stage of the recursion. Clearly show the final state of the array A.
- Write and justify the runtime recurrence for the above algorithm.
- Use Master Method to theta-value for the runtime of this algorithm.

Question 4: Finding the runtime from recurrence relations

(a) 12 points Use the recursion tree to find a tight asymptotic bound for each of :

- $T(n) = 2T(n/4) + 1$
- $T(n) = 4T(n/4) + n$
- Show $T(n) = T(n/2) + \log n$ is $O((\log n)^2)$

- $T(n) = 9T(n/3) + n^2$

**Reminder to refer to the summation formulas on our math sheet.*

(b) 8 points Apply the master theorem to to each of the following, or state that it does not apply:

- $T(n) = 2T(n/2) + n^2 + n$
- $T(n) = 15T(n/4) + n^2 \log n$
- $T(n) = 17T(n/4) + n^2 + \log n.$
- $T(n) = 16T(n/4) + n^2 \log n + n^3$